


Towards Real-Time Classification of HPC Workloads via Out-of-band Telemetry

Steven Presser
HLRS, University of Stuttgart
Stuttgart, Germany
steven.presser@hls.de 

Abstract—Detecting illicit workloads on High Performance Computing (HPC) systems is an important task. Such workloads might indicate a user account is being misused, for example to run cryptocurrency miners or password crackers. Existing solutions use in-band collection of data, which may slow down the workloads on the system. We present early results demonstrating that real-time classification of HPC workloads based on out-of-band telemetry may be possible. Further, we show a method-dependent accuracy of greater than 99%, with the best results achieving near-perfect classification. These results suggest that creating a real-time classifier of HPC jobs that does not impact system performance is possible.

Index Terms—HPC, Computer Security, Supercomputers, Artificial Intelligence, Application detection Monitoring

I. INTRODUCTION

Due to their immense computing power, High Performance Computing (HPC) systems are targets for misuse, both by external actors [1] and internal users [2]–[4]. Maintaining the security of these systems is critical to nations’ ability to forecast weather, predict the spread of disease, and predict the movements of refugees, among other uses. In 2020, a series of attacks [1] forced several supercomputers offline early in the COVID-19 pandemic, preventing their use for pandemic forecasting and demonstrating how critical these systems are to national security.

Security controls in HPC are complicated to implement, as “[T]here is a reluctance . . . to agree to any solution that might impose overhead on the system” [5]. Therefore, HPC systems often have significant security protecting the outside of the system (*e.g.*: network border, user accounts), but relatively little within the system itself. Currently, there is no solution which allows real-time system-wide monitoring of running HPC workloads for illicit behavior without significant negative impacts on performance.

II. BACKGROUND

An HPC system (often called a supercomputer or a super-computing cluster) is a set of computers (called nodes) which can be assigned to users to run user-specified software. Typically, they have a high-performance network interconnect (*e.g.*: 100GiB+ Ethernet, InfiniBand, Slingshot) and some large-volume storage accessible to all nodes in the system (*e.g.*: a Lustre appliance or Ceph filesystem). They may also have fast local storage to use as a cache (*e.g.*: NVME storage) and/or accelerators (*e.g.*: Graphical Processing Units (GPUs), Field

Programmable Gate Arrays (FPGAs), or Tensor Processing Units (TPUs)).

The individual nodes are typically assigned work by a batch-scheduling system. A user specifies how many nodes, how much time they need and what software to run. The batch system then assigns the nodes, which then run the software. The software the user runs is referred to as the workload or the job.

HPC systems have many layers of security protecting them. Many use the ScienceDMZ architecture [6], which allows systems administrators significant flexibility in adjusting the security measures to the risk tolerance and needs of the site.

However, HPC users will not tolerate any security measures that significantly affect the performance of workloads on the system – anything which causes “excessively high energy consumption or computational performance overhead” [5] is unacceptable. Therefore, in practice, many HPC systems rely primarily or entirely on the security of the network around them.

One area of active research is the detection of malicious action or workloads. Some of this work is based on user behavior [7], while others examine the HPC workloads themselves. One approach is to classify HPC workloads into categories. These might be “malicious” and “not malicious” or more specific to the software being run on the system. Primarily, workload classification has been performed on specific technologies, for example MPI [8]–[11]. However, work has also been done on classification via power signatures [12], and on general system-wide performance counters [13]. All of this previous work relies on modifying the HPC system, either by adding software (and therefore overhead) or by modifying the hardware itself. Both make such solutions problematic or impractical in the real world. Additionally, much prior work focuses on classifying workloads after they are complete, which is of little use for detecting ongoing security issues.

In practice, this means that HPC systems administrators have very few options for detecting if the software running on their systems is illicit: for example, forbidden by site policy, malicious, or may have been run by someone who has illegitimately logged in as a legitimate user.

One as-yet-unexplored area is the use of out-of-band telemetry. Out-of-band telemetry is gathered from the compute hardware itself, and does not require running any software in the operating system. This means data collection would

have no impact on the performance of the workload. Primarily out-of-band data is collected via the baseboard management controller (BMC). The BMC is included in server/datacenter class hardware and typically provides remote access, power control, health monitoring and telemetry. From this interface, a wealth of information is available, including:

- Processor metrics (*e.g.*: current speed, instructions per second)
- Memory access statistics
- Sensor readings (often including temperature)
- Network packet counts
- Disk access statistics

Many manufacturers present this information via RedFish [14], an industry-standard interface.

The remainder of this paper asks whether it is possible to perform classification of running HPC workloads based on out-of-band telemetry. If so, such classification could allow real-time detection of illicit workloads on HPC systems without any negative performance impact.

III. MOTIVATION

Much prior work exists on classifying HPC jobs, some focusing on security purposes. However, as previously mentioned, existing work runs once workloads are finished and/or using in-band metric data. Such systems are of limited utility from a security point of view because they can only detect security breaches after the fact. Further, they impose overhead on the HPC system itself, slowing and reducing the utility of legitimate HPC workloads.

In order to deploy a system to detect illicit HPC workloads in the real world, such a system must meet two goals. First, it must not impact the performance of the running workload. This goal can be met by using out-of-band data sources. Second, it must be able to detect illicit workloads while they are running. That is, they must operate in real-time and on workloads that are still in progress.

If both of these goals are met, this addresses the problems of previous work and makes classification of HPC workloads practical as a security tool.

IV. METHODOLOGY

This paper evaluates the feasibility of performing real-time classification of HPC workloads based on out-of-band telemetry data. At the moment, no public dataset of out-of-band workload telemetry is available. Therefore, we first synthesize such an out-of-band dataset from an existing in-band dataset. We then perform classification on the jobs in the synthesized dataset and compare results with the original dataset. Thus, we determine if classification of HPC jobs based on out-of-band data is feasible without significant loss of precision versus in-band data. Next, we examine two possible methods for performing classification of in-progress (*e.g.*: not yet completed) HPC workloads, collecting data on both correctness and runtime. This data is then examined to determine if real-time classification of HPC workloads (*e.g.*: performing classification as data is available without queuing

or dropping data) is feasible and what scale of resources may be required.

A. Dataset Synthesis

We selected the Taxonomist dataset [13], [15] as a base dataset. The Taxonomist dataset is a set of metrics collected with LDMS [16] on Volta, a Cray XC30m at Sandia National Laboratories. The dataset includes data for 11 benchmark workloads. It does not include data for cryptocurrency or password cracking workloads. This dataset is well-suited as a base because it is representative of typical HPC workloads running on an HPC system. Further, it collects a large number of metrics from a variety of data sources, such that a portion of the available data will be similar to what could be collected via out-of-band mechanisms.

We select the RedFish [14] protocol as representative of out-of-band telemetry collection protocols. While relatively new, RedFish has already seen adoption in industry, for example by major manufacturers of HPC hardware, such as in HPE's datacenter products [17], Intel datacenter products [18], SuperMicro datacenter products [19], and Lenovo datacenter products [20]. It has also seen adoption in HPC systems, including Cray's Shasta Systems Management software [21]. Because of the wide adoption of RedFish, we believe it represents a reasonable baseline for the information available via out-of-band controllers.

The Taxonomist dataset provides significant data that is not provided by the RedFish protocol. It must therefore be transformed before it may be used to represent the data stream available via an out-of-band controller. After examining the data available via RedFish and examining the data available in the Taxonomist dataset, we determined that the following metrics were available in the Taxonomist dataset and via out-of-band telemetry:

- Percent of time in kernel mode
- Percent of time in user mode
- Transmitted network frames
- Blocks allocated to cache
- Free memory (in MB)
- Disk bytes read
- Power consumption (Watts)

Therefore, this data was retained and all other data was discarded.

B. Taxonomist Comparison

Our experiment began by verifying the dataset synthesis step did not cause a significant loss of precision when classifying data. In order to do so, we compared the results of classification using the synthesized dataset to those in Ates [13], which used the Taxonomist dataset to perform classification based on in-band data. We began by performing identical feature engineering to Ates [13]. For each metric, we calculated the following features over each HPC workload:

- Maximum
- Minimum
- Mean

- Standard deviation
- Skew
- Kurtosis
- 5th percentile
- 25th percentile
- 50th percentile
- 75th percentile
- 95th percentile

The data was then spit into training and validation datasets, with the validation set comprising 20% of the total data.

The training set was used to train the same classifiers as in Ates [13]. Each classifier was then applied to the validation set. To match the results in Ates, any classification with a confidence of lower than 75% was labeled as "Unknown". We then calculated precision, recall, and f-score as in Ates.

C. Partial Workloads and Real-time Evaluation

Next, we evaluated classification of in-progress HPC workloads and determined if such classification can be performed in real time. We evaluated several classification methods as well as two methods for generating input vectors for classification.

First, the methods for generating input vectors. Both methods engineered the same features as used earlier.

The first method, here called Cumulative Classification, uses the entire history of the job to the current point in time to calculate the features. This closely matches the earlier comparison with Ates [13] and allowed us to evaluate if classification of in-progress workloads is viable and how much runtime may be required before classification is viable. However, using this strategy on real-world HPC workloads – which may run for days – might be expensive in terms of storage and computation.

The second method, here called Rolling Classification, calculates the features based on a rolling window. This strategy would likely be less computationally and storage intensive. It may also be better able to classify HPC workloads which change profile significantly or which have illicit software which only runs part of the time. However, it is possible that there may be a degradation in classification performance due to the relatively short window.

Second, classification methods. The specific classification methods were selected based on the best performing classifiers in the Taxonomist comparison. The comparison evaluated methods from two major families: Tree-based and Support Vector Machine (SVM)-based. Those methods which performed as well using out-of-band data as in-band data were selected for evaluation here.

V. RESULTS

A. Comparison to Taxonomist Results

In order to validate that the synthesized out-of-band data can be used for classification, we first compare the results of classifying single data points of synthesized out-of-band data representing entire jobs to the in-band results in Ates [13].

The results are shown in Table I.

TABLE I
COMPARISON OF RESULTS BETWEEN IN-BAND DATA AND SYNTHESIZED OUT-OF-BAND DATA. VALUES FOR IN-BAND ARE FROM ATEs [13]

Classifier	Data Type	Precision	Recall	F-Score
Random Forest	<i>In-band</i>	1.000	1.000	1.000
	<i>Out-of-band</i>	1.000	1.000	1.000
Extra Trees	<i>In-band</i>	1.000	1.000	1.000
	<i>Out-of-band</i>	1.000	1.000	1.000
Decision Tree	<i>In-band</i>	0.998	0.998	0.998
	<i>Out-of-band</i>	1.000	1.000	1.000
LinearSVC	<i>In-band</i>	0.999	0.999	0.999
	<i>Out-of-band</i>	0.987	0.904	0.942
SVC (RBF kernel)	<i>In-band</i>	0.994	0.994	0.994
	<i>Out-of-band</i>	0.997	0.959	0.997

These results show that the classification of the synthesized out-of-band data is as good or better than that of the in-band data for tree-based classifiers (RandomForest, ExtraTrees, DecisionTree). The performance with synthesized out-of-band data is as good as the original in-band data for the Random Forest and Extra Trees classifiers, and better for the Decision Tree classifier.

For SVC-based methods the results are more mixed. The linear SVC performs clearly worse on out-of-band data. However, the SVC using an RBF kernel shows a marginally better precision, but much reduced recall. Overall this implies the SVC classifiers perform worse on the synthesized out-of-band data. This may be due to the reduced breadth of data available to these methods.

Overall, this shows that the synthesized out-of-band data is still viable for classification, although the use of out-of-band data may change which classifiers are most suitable. In turn, this demonstrates that classification of HPC workloads based on out-of-band data collection is possible.

B. Cumulative Classification

In cumulative classification, we examine classification results for the engineered features over all previous data collected for each job for each sampling window. This simulates a system where, as each new sample comes in, the engineered features are recalculated and classified.

We exclude SVC-based methods from evaluation here for two reasons. First, they performed poorly with synthesized out-of-band data, as compared to their results with in-band data. Second, the training time for SVC-based methods grows exponentially with the number of samples [22]. The cumulative and rolling data engineering methods both create many input samples: about 2 million samples when based on the Taxonomist dataset. This results in exceptionally long training time. Further, deployment to a supercomputer could require significantly larger datasets, making the SVC-based methods unsuitable for use in this scenario. Therefore, they are eliminated from evaluation here.

Results for the remaining classifiers are shown in Table II.

These results exclude only the first and last 40 seconds of runtime (as in Ates [13]). Interestingly, they show excellent

TABLE II
RESULTS OF VARIOUS CLASSIFIERS FOR CUMULATIVE ENGINEERED FEATURES

Classifier	Precision	Recall	F-Score	Classifications per Second
Random Forest	1.000	0.996	0.998	18,984
Extra Trees	1.000	0.999	0.999	20,336
Decision Tree	0.999	0.999	0.999	317,722

results, indicating that classification is largely successful, even very early in workloads.

C. Rolling Classification

Next, we examine classification results for a rolling 40-sample window of the synthesized out-of-band data. The engineered features are calculated for each 40-sample window, then classified. The results are shown in Table III.

TABLE III
RESULTS OF VARIOUS CLASSIFIERS FOR ROLLING 40-SAMPLE WINDOWS

Classifier	Precision	Recall	F-Score	Classifications per second
Random Forest	1.000	1.000	1.000	18,796
Extra Trees	1.000	0.999	0.999	18,058
Decision Tree	0.996	0.996	0.996	311,165

These results show near-perfect classification for Random-Forest and ExtraTrees classifiers, with additional excellent results for DecisionTrees. This shows there is not a significant loss of accuracy when using a window, versus classification of cumulative data or versus classifying entire workloads.

VI. DISCUSSION

The results shown here demonstrate three important prerequisites for performing real-time classification of HPC workloads based on out-of-band data.

First, they demonstrate that classification of HPC workloads via data collected out-of-band is possible with minimal or no loss of accuracy. This is shown via a comparison of classification results with the synthesized out-of-band dataset versus the Ates [13] results in Table I. In that comparison, tree-based classifiers using simulated out-of-band data performed as well as or better than tree-based classifiers using the original in-band data.

Second, these results show that classification can be performed on running HPC workloads with a high degree of accuracy, even though the job is still in-progress. Much prior work has focused on classification of HPC jobs after they are complete. However, this approach is of little use in detecting ongoing threats or reacting to security issues in progress. Therefore, showing that classification is viable on ongoing/in-progress workloads is an important step towards making HPC workload classification a practical security tool.

Surprisingly, our results show rolling classification to be marginally more accurate than cumulative classification. We theorize this is because in rolling classification any outlier values in any metric relatively quickly move outside the window and do not affect later classifications, whereas in cumulative classification the outlier remains in the data until the workload finishes.

Third, measurement of classification throughput shows that classification of HPC workloads can be performed in real time. The throughputs measured here range from 18,058 classifications per second per core to 317,722 classifications per second per core. One classification corresponds to one source of out-of-band data, which corresponds to one node within the HPC system. While HPC system node counts vary widely, Table IV demonstrates that a single system with a relatively modest core count would likely be sufficient to perform classification for some of the world’s largest HPC systems, including HAWK, located at HLRS, our home institution.

TABLE IV
HPC NODE COUNTS AND REQUIRED CORES FOR CLASSIFICATION. ALL VALUES CALCULATED USING 18,000 CLASSIFICATIONS PER SECOND PER CORE.

System Name	June 2022 Top 500 Rank [23]	Nodes	Cores required for classification once per second
Frontier	1	9,408 [24]	1
Fugaku	2	158,976 [24]	9
LUMI	3	2560 [25]	1
Summit	4	4,356 [23]	1
Sierra	5	4,320 [23]	1
HAWK	27	5,632 [26]	1

However, real-world performance will likely be worse than these estimates. The estimates exclude overhead in the classification system for receiving data from out-of-band node controllers, from calculating the vectors that are input to the classifiers, and from interpreting the results - though all this work could be performed on other cores within the system performing the out-of-band analysis.

We would also make changes to the classifiers for use in a real-world system. For example, we held the confidence threshold at 75% in all evaluations. For a dataset in which each point should receive a label, this is acceptable. However, a real world system would need to deal with workloads of an unknown type, where this may not be appropriate. Additionally, a real-world system would prioritize correct and timely classification of workloads – a different balance than correctly classifying individual data points.

VII. FUTURE WORK

Our results demonstrate that machine learning on out-of-band HPC telemetry is sufficient to classify the workloads running on an HPC system in real-time. However, the results are built on a dataset collected in-band, not out-of-band. Additionally, the dataset is running benchmarks, not real HPC workloads. Thus, while these results demonstrate the

feasibility of real-time classification of HPC workloads based on out-of-band telemetry, significant future work remains to demonstrate this using data collected out-of-band.

Therefore, future work will begin by collecting real out-of-band data from a real HPC system running real workloads. It will then demonstrate that machine learning is still an effective method for classifying workloads and will demonstrate that the classification may be performed in real-time. Further, it will evaluate the resources required for classification on a real-world system.

Additionally, the dataset we used to synthesize out-of-band data did not include any examples of illicit workloads. Therefore, future work must build a dataset which includes such workloads and evaluate classifier performance for illicit workloads.

VIII. CONCLUSION

This work demonstrates the viability of performing real-time classification of in-progress HPC workloads, using out-of-band data. We achieve excellent results, with some classifiers performing with near-perfect accuracy. This is an important problem, as performing such classification would allow detection of and reaction to illicit workloads on HPC systems (such as password crackers, cryptocurrency miners or similar software) in real time. Prior solutions use in-band data or require modification of hardware, limiting their utility. The use of out-of-band data allows monitoring of the system to occur without impact on the performance of the HPC system. Real-time monitoring based on out-of-band data would significantly increase the security of HPC systems and ensures these critical national security assets are available for everything from weather prediction to pathogen modeling and beyond.

REFERENCES

- [1] "Hacking Streak Forces European Supercomputers Offline in Midst of COVID-19 Research Effort." <https://www.hpcwire.com/2020/05/18/hacking-streak-forces-european-supercomputers-offline-in-midst-of-covid-19-research-effort/>, May 2020.
- [2] B. Cohen, "US Government Bans Professor for Mining Bitcoin with A Supercomputer." <https://bitcoinmagazine.com/business/government-bans-professor-mining-bitcoin-supercomputer-1402002877>.
- [3] "Harvard Research Computing Resources Misused for 'Dogecoin' Mining Operation." <https://www.thecrimson.com/article/2014/2/20/harvard-odyssey-dogecoin/>.
- [4] "Russian Nuclear Engineers Caught Cryptomining on Lab Supercomputer." <https://www.hpcwire.com/2018/02/12/russian-nuclear-engineers-caught-cryptomining-lab-supercomputer/>, Feb. 2018.
- [5] S. Peisert, "Security in High-Performance Computing Environments." <https://cacm.acm.org/magazines/2017/9/220422-security-in-high-performance-computing-environments/fulltext>.
- [6] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski, "The Science DMZ: A network design pattern for data-intensive science," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, (Denver Colorado), pp. 1–10, ACM, Nov. 2013.
- [7] T. Miles-Board, "Bearcade: A Novel High-Performance Computing User and Security Management System Augmented with Machine Learning Technology," p. 183.
- [8] S. Peisert, "Fingerprinting Communication and Computation on HPC Machines," Tech. Rep. LBNL-3483E, 983323, June 2010.
- [9] S. Whalen, S. Peisert, and M. Bishop, "Multiclass Classification of Distributed Memory Parallel Computations," *Pattern Recognition Letters (PRL)*, vol. 34, pp. 322–329, Nov. 2012.
- [10] S. Whalen, S. Engle, S. Peisert, and M. Bishop, "Network-theoretic classification of parallel computation patterns," *The International Journal of High Performance Computing Applications*, vol. 26, pp. 159–169, May 2012.
- [11] O. DeMasi, T. Samak, and D. H. Bailey, "Identifying HPC codes via performance logs and machine learning," in *Proceedings of the First Workshop on Changing Landscapes in HPC Security - CLHS '13*, (New York, New York, USA), p. 23, ACM Press, 2013.
- [12] J. Combs, J. Nazor, R. Thysell, F. Santiago, M. Hardwick, L. Olson, S. Rivoire, C.-H. Hsu, and S. W. Poole, "Power Signatures of High-Performance Computing Workloads," in *2014 Energy Efficient Supercomputing Workshop*, pp. 70–78, Nov. 2014.
- [13] E. Ates, O. Tuncer, A. Turk, V. J. Leung, J. Brandt, M. Egele, and A. K. Coskun, "Taxonomist: Application Detection Through Rich Monitoring Data," in *Euro-Par 2018: Parallel Processing* (M. Aldinucci, L. Padovani, and M. Torquati, eds.), Lecture Notes in Computer Science, (Cham), pp. 92–105, Springer International Publishing, 2018.
- [14] "Redfish Specification." https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.15.0.pdf.
- [15] "Artifact for Taxonomist: Application Detection through Rich Monitoring Data," Aug. 2018.
- [16] A. Agelastos, B. Allan, J. Brandt, P. Cassella, J. Enos, J. Fullop, A. Gentile, S. Monk, N. Naksinehaboon, J. Ogden, M. Rajan, M. Showerman, J. Stevenson, N. Taerat, and T. Tucker, "The Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications," in *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 154–165, Nov. 2014.
- [17] "ILO RESTFUL API ECOSYSTEM." <https://www.hpe.com/us/en/servers/restful-api.html>.
- [18] "Intel Platform Service Assurance – Retrieving Redfish* Platform Telemetry Feature Brief," p. 2.
- [19] "Redfish@ API — Supermicro." <https://www.supermicro.com/en/solutions/management-software/redfish>.
- [20] "Lenovo XClarity Controller REST API Guide," p. 238, 2018.
- [21] S. J. Martin, D. Rush, K. Hughes, and M. Kelly, "Modernizing Cray R Systems Management Use of Redfish R APIs on Next Generation Cray Hardware," p. 10.
- [22] "1.4. Support Vector Machines." <https://scikit-learn/stable/modules/svm.html>.
- [23] "June 2022 — TOP500." <https://www.top500.org/lists/top500/2022/06/>.
- [24] "Top500: Exascale Is Officially Here with Debut of Frontier." <https://www.hpcwire.com/2022/05/30/top500-exascale-is-officially-here-with-debut-of-frontier/>, May 2022.
- [25] "LUMI's full system architecture revealed," Nov. 2021.
- [26] "Batch System PBSPro (Hawk) - HLRS Platforms." [https://kb.hlrs.de/platforms/index.php/Batch_System_PBSPro_\(Hawk\)#Node_types](https://kb.hlrs.de/platforms/index.php/Batch_System_PBSPro_(Hawk)#Node_types).